Introduction to Python & Data

Research Data Management Support

02-03-2023

Table of contents

Welcome!

Python is a powerful programming language that is popular can be used to write scripts for to work effectively and reproducibly with data. In this workshop, we aim to give you the tools to start exploring Python and all it has to offer by yourself.

1 Acknowledgements

Our course materials are an adaptation of the …. Carpentries course, reused under a CC-something license. The Intro to R & Data course at UU also served as a reference.

1.1 Contributors

- Jelle
- Christine
- Roel
- Neha
- Stefano

2 Schedule

we can add a detailed schedule here later

3 Installation & Setup

Hello plz don’t download Python from the Microsoft Store :P

4 Course Materials

You can download the course materials as individual files or as zip files

- insert links to datasets and notebooks and zip files etc. below

5 Introduction

What is Python? Why do we like it so much more than R? Let's get going!

6 Introduction to python

6.1 Some simple statements

```
2+2
```

```
4
```

```
print("Hello World")
```

```
Hello World
```

6.2 Variables, values and their types

```
text = "Data Carpentry"
number = 42
pi_value = 3.141592653589793238462643383279502884197
```

```
text
```

```
'Data Carpentry'
```

```
type(text)
```

```
str
```

```
number
```

```
42
```

```
type(number)
```

```
int
```

```
pi_value
```

```
3.141592653589793
```

```
type(pi_value)
```

```
float
```

6.3 Output versus printing

In this example we first print the number and then call the variable again:

```
print(number)
number
```

```
42
```

```
42
```

Now we do it the other way around:

```
number
print(number)
```

```
42
```

The interpreter does not output the value of the variable unless it is the very last line in an input field. In general `print` is the only way to print output to the screen when you are not working in an interactive environment as Jupyter, but when you are working with scripts. Rule of thumb: use output for quick checking while developing your Jupyter notebook, use print for all output that needs to be there while running a Jupyter notebook.

```
summing = 2 + 2
multiply = 6 * 7
power = 2 ** 16
modulo = 13 % 5

print("Sum: ", summing)
print("Multiply: ", multiply)
print("Power: ", power)
print("Modulo: ", modulo)
```

```
Sum:  4
Multiply:  42
Power:  65536
Modulo:  3
```

6.4 Logical values, operators and variables

There are two logical values, *true* and *false*. Inpython they are decoded as the values `True` and `False`. Note: here these two are really distinct values and no strings!

```
True
```

```
True
```

```
False
```

```
False
```

With the logical operators `>`, `<`, `==`, `and`, `or` and `not` we can now compare variables and create logical statements.

```
compare = 3 > 4
print("3 > 4 : ", compare)
```

```
3 > 4 :  False
```

```python
not_compare = not compare
print("not(3 > 4): ", not_compare)
```

```
not(3 > 4):  True
```

```python
compare or not_compare
```

```
True
```

```python
compare and not_compare
```

```
False
```

```python
True or False
```

```
True
```

```python
True and False
```

```
False
```

```python
True == compare
```

```
False
```

6.5 The if-statement

We saw how to create logical statements. In the if-clause we use these as conditional statements to carry some tasks.

6.6 Lists and Tuples

```python
numbers = [1, 2, 3]
numbers[0]
```

```
1
```

```python
type(numbers)
```

```
list
```

```python
len(numbers)
```

```
3
```

```python
numbers[3]
```

```
IndexError: list index out of range
```

```python
numbers[-1]
```

```
3
```

```python
numbers[2] == numbers[-1]
```

```
True
```

```python
words = ["cat", "dog", "horse"]
words[1]
```

```
'dog'
```

```python
type(words)
```

```
list
```

```python
if type(words) == type(numbers):
    print("these variables have the same type!")
```

```
these variables have the same type!
```

```python
newlist = ["cat", 1, "horse"]
```

```python
type(newlist[0])
```

```
str
```

```python
type(newlist[1])
```

```
int
```

```python
numbers.append(4)
print(numbers)
```

```
[1, 2, 3, 4]
```

```python
numbers[2] = 333
print(numbers)
```

```
[1, 2, 333, 4]
```

```
# Tuples use parentheses
a_tuple = (1, 2, 3)
another_tuple = ('blue', 'green', 'red')

# Note: lists use square brackets
a_list = [1, 2, 3]
```

```
a_list[1] = 5
print(a_list)
```

```
[1, 5, 3]
```

```
a_tuple[2] = 5
print(a_tuple)
```

```
TypeError: 'tuple' object does not support item assignment
```

```
type(a_tuple)
```

```
tuple
```

6.7 Dictionaries

```
my_dict = {'one': 'first', 'two': 'second'}
my_dict
```

```
{'one': 'first', 'two': 'second'}
```

```
my_dict['one']
```

```
'first'
```

```python
my_dict['third'] = 'three'
my_dict
```

```
{'one': 'first', 'two': 'second', 'third': 'three'}
```

6.8 For loops

```python
numbers = [5, 6, 7]
```

```python
for item in numbers:
    print(item)
```

```
5
6
7
```

```python
words = ["cat", "dog", "horse"]
for index, item in enumerate(words):
    print(index)
    print(item)
```

```
0
cat
1
dog
2
horse
```

```python
for key, value in my_dict.items():
    print(key, '->', value)
```

```
one -> first
two -> second
third -> three
```

6.9 Exercises

6.9.1 Exercise 0

Try to run this code. Why is there no output?

```
x = 6
apple = "apple"
```

6.9.2 Exercise 1

1. Calculate: One plus five divided by nine.
2. Assign the result of the calculation to a variable.
3. Test if the result is larger than one.
4. Round off the result to one decimal. Use the function `round`.

```
s = (1+5)/9
round(s, 1)
```

```
0.7
```

6.9.3 Exercise 2

Predict the results:

```
5 == 5
not 3 > 2
True == 'True'
False < True
```

```
True
```

6.9.4 Exercise 3

Meet Ann, Bob, Chloe, and Dan. 1. Create a list with their names. Save the list as “name”.

2. How old are Ann, Bob, Chloe, and Dan? You decide! Design a numeric list with their respective ages. Save it as “age”.
3. What is their average age? (Use the function `sum` to sum up their cumulative ages, you can use `len(age)` to get the number of elements in a list)

```
name = ["Ann", "Bob", "Chloe", "Dan"]
age = [3, 30, 41, 2]
sum(age)/len(age)
```

```
19.0
```

6.9.5 Exercise 4

1. Return only the first number in the list age.
2. Return the 2nd and 4th name in your list name.
3. Return only ages under 30 from your list age.
4. Return the name “Chloe” from the list name.

```
age[0]
[age[1], age[3]]
new_age=[]
for a in age:if a < 30:new_age.append(a)[n for n in name if n == "Chloe"] #or
name[name.index("Chloe")]
```

```
SyntaxError: invalid syntax (4124803616.py, line 4)
```

6.9.6 Exercise 5

Make an if statement that tests if a number is even, and saves the classification in a variable called number_class.

```
number = 5
if number%2 == 0:
    number_class = "even"
else:
    number_class = "odd"
print(number, "is", number_class)
```

```
5 is odd
```

6.9.7 Exercise 6

Turn the if statement from the last exercise into a function. Let the user provide the value for number, and return the number_class.

```
def even_or_odd(number):
    if number%2 == 0:
        number_class = "even"
    else:
        number_class = "odd"
    return number_class
```

```
number = 5
print(number, "is", even_or_odd(number))
```

```
5 is odd
```

6.9.8 Exercise 7

Use the function above to determine whether the numbers between 1 and 10 are even or odd.

```
for i in range(1, 11):
    res = even_or_odd(i)
    print(i, 'is', res)
```

```
1 is odd
2 is even
3 is odd
4 is even
5 is odd
6 is even
7 is odd
```

```
8 is even
9 is odd
10 is even
```

7 Data Science with Pandas

Let's try out the Pandas library!

```
2 + 2
```

```
4
```

```
print("Hello World")
```

```
Hello World
```

This is a book created from markdown and executable code.

See Knuth (1984) for additional discussion of literate programming.

References

Knuth, Donald E. 1984. "Literate Programming." *Comput. J.* 27 (2): 97–111. https://doi.org/10.1093/comjnl/27.2.97.